5.3 ACT-UP: A Toolkit for Hampton, Cognitive Modeling Composition, Reuse and Integration

ACT-UP: A Cognitive Modeling Toolkit for Composition, Reuse and Integration

Christian Lebiere and David Reitter
Carnegie Mellon University
cl@cmu.edu; reitter@cmu.edu

- Computational implementation of unified theory of cognition
- Commitment to task-invariant mechanisms
- Modular organization
- Limited capacity
- Hybrid symbolic statistical processes

ACT-R Cognitive Architectures
Motivations and Applications

- **Philosophy**: Unified understanding of the mind.
- **Psychology**: Account for experimental data.
- **Education**: Provide cognitive models for intelligent tutoring systems and other learning environments.
- **Human Computer Interaction**: Evaluate artifacts and help in their design.
- **Computer Generated Forces**: Provide cognitive agents to inhabit training environments & games.
- **Neuroscience**: Provide a framework for interpreting data from brain imaging.

Goals

- Enable the implementation of more complex ACT-R models
- Scale up cognitive models to simulate learning / adaptation in communities (e.g., about 1,000 models in parallel)
- Treat models as hard claims
  - Evaluate each specified component against data
  - Underspecify the rest and fit free parameters
The Argument

- **Constraints:** Architectural advances require further constraints

- **Scaling it up:** Complex tasks, broad coverage of behavior (e.g., linguistic), use of microstrategies and predictive modeling may serve to motivate further architectural constraints

- **Difficulties:** ACT-R is heavily constrained already, and models are difficult to develop, reuse and exchange

Control Structure

A flow-chart describes an algorithm (or a cognitive strategy)

- Decision-making points and states
- Not easy to reuse: it fails to capture generalizations

Computer Science: pre-Object Orientation, pre-Functional Programming
Decomposition

Production Rule System
The Argument

• **Constraints:** Architectural advances require further constraints

• **Scaling it up:** Complex tasks, broad coverage of behavior (e.g., linguistic), use of microstrategies and predictive modeling may serve to motivate further architectural constraints

• **Difficulties:** ACT-R is heavily constrained already, and models are difficult to develop, reuse and exchange

• **We need to produce models at a higher abstraction level**
  – However, we’d like to leverage successful cognitive modules, describing memory retention, cue-based retrieval, routinization, reinforcement learning

Cognitive Strategy

Subsymbolic (Learning / Adaptation)

Symbolic

deterministic

non-deterministic

explains empirical variance
Priming Model

Crucial request of a chunk from declarative memory

• Only a small portion of the model explains the behavioral data at hand
• The rest explains that the task can be accomplished in principle with a parallel architecture and with specific cognitive representations (chunk types)

Production Systems vs. assembly language

evensum: cir.1 D1 ;Zero-out
sumloop: add.1 D0, D1 ;Add current ;counter value to
accumulator
subq #1, D0 ;Decrement
bne sumloop ;until it ;reaches zero
mul 2, D1 ;Double sum to account ;for even numbers
rts ;Return ;to caller

~1990
The Argument

- **Constraints**: Architectural advances require further constraints
- **Scaling it up**: Complex tasks, broad coverage of behavior (e.g., linguistic), use of microstrategies and predictive modeling may serve to motivate further architectural constraints
- **Difficulties**: ACT-R is heavily constrained already, and models are difficult to develop, reuse and exchange
- **Abstraction**: To implement those, we need to produce models at a higher abstraction level
- **Underspecification is the key to focus on verifiable claims, and to avoid overfitting by fitting free parameters to data**

Underspecified Models
ACT-UP

- A stand-alone system on the basis of Common Lisp
- targets an audience that can write simple Lisp programs (unlike, e.g., CogTool)
- Toolbox approach to ACT-R
  - light-weight: it's a Lisp library
  - does not produce production rules (ACT-R/Lisa, ACT-Simple, CogTool)
- Not aimed at implementing all constraints of ACT-R 6 (unlike Java ACT-R, Python ACT-R)

Declarative Memory

- `define-chunk-type`
  - types are optional
- `make-count-order`
- `learn-chunk`
- `defrule`
- `retrieve-chunk`
- `count-order-second`
ACT-UP is not ACT-R 6...

- ACT-UP Interface is synchronous
  - Serial execution
  - Deterministic strategies defined as programs
- Parallelism (e.g., perceptual/motor modules) possible [not implemented]
- Non-deterministic rule choice is possible
  - Reinforcement-learning as in ACT-R 6

PM / Utility learning

- `choose-coin`
- Calls either `decide-heads` or `decide-tails`
- `assign-reward` reinforces the decision
- Exact production rules are underspecified, but decision-making point is explicit
- Choice model replicates ACT-R and empirical results

```
;; Experimental environment
(defun toss-coin ()
  (if (< (random 1.0) .9) 'heads 'tails))

;;; The Model
;;; Rules that return the choice as symbol heads or tails
(defrule decide-tails ()
  (group choose-coin 'tails))

(defrule decide-heads ()
  (group choose-coin 'heads))
```
Debugging

(defun debug-detail (do-it i)
  (debug-match-chunk (lookup-type*): No such chunk in DM. Returning new chunk (not in DM) of name LOST
  Presentation of chunk LOST (VP: NIL, t=72761.06. H: MODELS/496, b=0.
  Implicitly creating chunk of name LOST.
  Presentation of chunk HAD (VP: NIL, t=72761.445. H: MODELS/496, b=0.
  Implicitly creating chunk of name HAD.
  Presentation of chunk MB (VP: NIL, t=72761.3085. H: MODELS/496, t=72761.305.
  Implicitly creating chunk of name MB.
  Group PAST-TENSE-MODEL with 1=0 matching rules, choosing rule PMID (Utility 5.67890998)
  Group PAST-TENSE with 3=0 matching rules, choosing rule STRATEGY-WITHOUT-ANALOGY (Utility 5.223997)
  retrieve-chunk:
    spec: (CHUNK-TYPE PAST-TENSE VERB GET)
  curv: NIL
  past: NIL
  filtered 0 matching chunks.
  retrieved none out of 0 matching chunks.
  NIL
  Assigning reward 0.9
  Assigning reward 3.53325 to STRATEGY-WITHOUT-ANALOGY. STRATEGY-WITH-ANALOGY remains best regular rule in group PAST-TENSE-MODEL.
  Assigning reward 0.0 to PMID. Best regular rule among alternatives in group PAST-TENSE-MODEL!
  NIL

CL-USER>
Implemented Models

• 10 Classic models implemented:
  – count, addition, siegler, zbrodoff, paired, fan, sticks,
    semantic, choice, past-tense

* past-tense not yet complete

Efficiency

• Sentence production (syntactic priming) model
  – 30 productions in ACT-R, 720 lines of code
  – 82 lines of code in ACT-UP (3 work-days)
  – ACT-R 6: 14 sentences/second
  – ACT-UP: 380 sentences/second
Scalability

- Language evolution model
  - Simulates domain vocabulary emergence (ICCM 2009, JCSR 1010)
  - 40 production rules in ACT-R (could not prototype)
  - 8 participants interacting in communities
- In larger community networks: 1000 agents, 84M interactions (about 1 minute sim. time each), 37 CPU hours

Rapid prototyping/Reuse

- Dynamic Stocks&Flows model (JAGI 2010)
  - Competition entry, model written in < 1 person-month
  - Instance-based learning (IBL, Gonzales&Lebiere 2003)
  - Blending (Wallach&Lebiere 2003)
  - free parameters (timing) estimated from example data
  - Model generalized to novel conditions
    - (.... NOT. but it did so better than others.)
- Same IBL/blending micro-strategy was re-used directly in a Lemonade Stand Game entry to a 2009 competition (BRIMS 2010)
Drawbacks

- Less established code-base than ACT-R 6
- Lisp
- Lack of architectural timing predictions from rule matching
- Lack of parallelism (planned: fall 2010)
- Lack of perception/motor modules
  - Will be available in ACT/Simple-style interface
    (Salvucci&Lee 2003)

Beta-Test

- **Limited Release** of ACT-UP test version
  - comes with 10 example models
  - 4 tutorials (paralleling the ACT-R 6 ones)
  - Full API documentation plus How-do-I... document
- Testing period: Fall 2010
- Task: implement 1-2 models of your own
- Review letter requested (journal-review style)